Apr. 30
International Journal of advanced studies in Computer Science and Engineering
IJASCSE Volume 4, Issue 4, 2015

An Approach of Graph Isomorphism Detection based on Vertex-Invariant

Vijaya Balpande
Dept of Computer Science and Engineering
GHRCE,, PJLCOE, Nagpur, India

Anjali Mahajan
Dept of Computer Science and Engineering
PIET, Nagpur, India

***Abstract*— Graph Isomorphism is a widely studied problem due to its practical applications in various fields of networks, chemistry and finger print detection, recent problems in biology such as diabetes detection, protein structure and information retrieval. An approach to the graph isomorphism detection is based on vertex invariant. In the existing approach vertex invariants is used to partition the matrix of input graph for reducing size of decision tree to detect graph isomorphism. In this approach the graph isomorphism detection is carried out by comparing the entire adjacency matrix of input graph with the model graph. Element by element comparisons requires more time to detect graph isomorphism. In this paper we present graph isomorphism detection algorithm based on vertex invariant and Euclidean vector. We evaluate the proposed algorithm on the various randomly generated directed and undirected graphs by computing the Euclidean vector of input graph adjacency matrix and model graph to make comparison with existing algorithm. Experimental result shows that computational time complexity is reduced remarkably as compared to the existing sequential algorithm.**

Keywords*—Graph Isomorphism, Vertex Invariants, Eucledian vector

I. INTRODUCTION

Graphs are widely used in real life applications to represent the structure of objects, e.g. applications like molecules, images, networks Graph isomorphism is a way of matching the two graphs whether they are equivalent or not. There is complete structural equivalence between two graph and differs only in the names of vertices and edges. Graph Isomorphism is a way of one to one mapping of vertices and edges of two graphs. Graph isomorphism is highly studied problem in research field and graph theory. Graph isomorphism problem is extensively applied in many applications in various fields such as data mining [1], pattern recognition [2], information retrieval [3], chemistry [4].Most researchers believe that GI problem is not NP-complete. As there is no polynomial solution for GI it is not known to be in P nor to be NP-complete.

Graph matching can be done in two ways. Given two graphs, checking whether two graphs are isomorphic or one graph is a subgraph, of the other graph. In second method, an input graph is matched with graphs present in a given database called model graph .If an input graph entire structure is matched with model graph then two graphs are said to be isomorphic and if substructure of a graph is matched it is said to be subgraph isomorphic. In [5] graph isomorphism detection is based on decision tree approach based on the methodology proposed by[6].The existing algorithm uses the vertex invariant property to reduce the search space but the time complexity is same as[6].Vertex invariants[7] are the attributes assigned to the vertex which do not change after performing graph isomorphism. The attributes such as label of vertex, label of edge, degree of vertex are known as vertex invariants. By applying vertex invariants property we can change the position of vertices within the same group not from different group. In the existing algorithm, the adjacency matrix of the model graph and input graph are compared for testing isomorphism .In our approach we applied the Euclidean norm on adjacency matrix and compared the Euclidean vector of the input graph against the model graph matrix which reduces the time complexity as compared to existing algorithm[5].

In this paper section II describes the basic definitions and notations. Section III describes the basic idea of the algorithm. In section IV modified algorithm is described. In Section V experimental results were given. Section VI specifies the conclusion.

II. BASIC DEFINITION

A. *Definition1:*

In[8],the graph isomorphism is expressed as : Given two graphs G1=(V1,E1) and G2=(V2,E2) ,if there exist 1 to 1 mapping function f from v1 to v2 such that (i , j) ∈ E1, if and only if (f(i) ,f(j)) ∈ E2. The function f is called an isomorphism from G1 to G2 .If the two graphs isomorphic to each other, it is denoted by G1 ≅ G2.

B. *Definition2:*

The identity matrix M of order n x n is represented as

$$m_{ij} = \begin{cases} 1, \text{ if } i = j \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

Where *m*, is an element of *M* on the i^{th} row and j^{th} column

C. *Definition3:*

A permutation matrix is obtained from the identity matrix by any row and column permutation. If M_1 and M_2 are the two matrices of graph G_1 and G_2 respectively M_1 is said to be

isomorphic to M_2 if there exists 1 to 1 mapping of function f from the rows of M_1 to rows of M_2 and from column of M_1 to column of M_2. The function f is called an isomorphism from M_1 to M_2. In other words, M_1 is isomorphic to M_2 if and only if there exists the permutation matrices P_1 and P_2 satisfying the following relation

$$M_1 = P_1 M_2 P_2 \quad (2)$$

III. EXISTING ALGORITHM

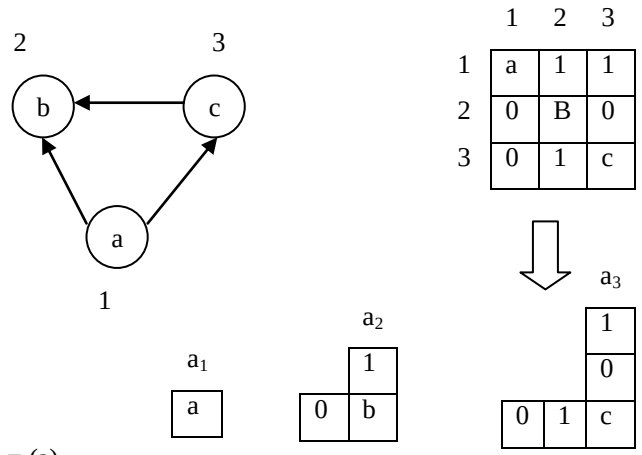
In [5], vertex invariant and decision tree concept is used to test graph isomorphism. The vertex invariants are used to partition the matrix of the graph before graph isomorphism detection. By using vertex invariant property of graph, the size of decision tree is reduced as compared to [6]. The technique is similar to breadth pruning technique which reduces the size of decision tree remarkably still the time complexity is almost equivalent.

Decision tree is the most and simple method for knowledge representation. The basic idea [6] of the isomorphism algorithm is that all possible permutation of adjacency matrix of each of the model graph was computed offline and the permutation matrices were represented as decision tree. The matrix of input graph is matched to those of adjacency matrices in the decision tree which are identical to it. At run time, the permutation matrices correspond to these adjacency matrices represent the graph or subgraph isomorphism.

By using the row-column element of each permutation matrix we can recognize the model graph into decision tree. A row-column element x_i of n x n matrix is a vector and is represented as $x_i = (y_{1i}, y_{2i} \cdots y_{ii}, y_{i\,(i-1)} \ldots . y_{i1})$. The representation of an adjacency matrix A by its row –column element is illustrated in figure 1. The x_1, x_2---- are the row column element of matrix x. A root node is present at the top of decision tree. At each level of the decision tree the classification is done by comparing the row column element of permutation matrix. For the first time, the classification is done by comparing the first row –column element of the input graph by the first row-column element x_i of each permutation matrix. At the n^{th} level of decision tree the classification is carried out by comparing the row-column element x_n of the permutation matrices. Graph G has 3 vertices and therefore it has 3! = 6 permutation matrices. The row column element of the 6 permutation matrices were then organized as a decision tree.

The decision tree formed is of exponential size depending on the number of vertices and requires huge amount of storage if the number of vertices increases. A graph with n vertices has n! permutation matrices. A row-column element at level n of decision tree would be n! at the worst case. In [5], vertex invariants are used to reduce the amount of permutation matrices which subsequently reduces the size of decision tree. As the size of decision tree is directly proportional to the permutation matrices. The vertex invariants are used to partition the vertices of graph into equivalence classes such that all the vertices assigned to the same partition have the same values for the vertex invariants. The size of the decision tree is reduced remarkably.

For the model and input graph, each of permutation matrices of input graph are computed and compared with the permutation matrices of the model graph. The comparison is carried out element by element. Time required for comparing the matrices is more as it needs to compare nxn elements of permutation matrices.



a_1= (a)

a_2 = (1, b, 0)

a_3 = (1, 0, c, 1, 0)

Figure 1: Row-column element

Input: Graph G = (V, E) represented by adjacency matrix
Graph (G1, G2)
Output: Graphs are Isomorphic begin

1. Parse Graph (G1,G2)
2. Check Graph Type, degree, nodes and labels
3. Create Permutations of partition graph
   Pg1 {{P1}, {P2} ---- {Pn}}
   Pg2 {{P1}, {P2} ---- {Pn}}
4. Create adjacency matrix M1,M2,----,Mn of G1using Pg1
5. Create adjacency matrix T1 of G2 using Pg2.
6. For i = 1 to n
   Check if Mi= T1
7. Detect graph are isomorphic

end

Algorithm 1: Existing Algorithm for graph isomorphism

IV. GRAPH ISOMORPHISM USING EUCLEDIAN VECTOR

In the modified algorithm A2, we are using the vertex invariant property like the existing algorithm to find the count of permutation matrices. In our approach instead of computing the permutation matrices for the entire graph we compute the count of permutation matrix .Based on the count of permutation matrices we are computing the only one sequence of permutation matrix. For this matrix , compute the Euclidean vector of the input graph and the model graph. If the Euclidean vector of the input graph matches with the Euclidean vector of the model graph , graph isomorphism is detected and the two graphs are said to be isomorphic to each other. Thus by comparing Euclidean vector of matrices we are reducing the time complexity as compared to existing algorithm.

Input: Graph G = (V, E) represented by adjacency matrix
Graph (G1, G2)
Output: Graphs are Isomorphic begin

1. Parse Graph (GP1,GP2)
2. Check Graph Type, degree, nodes and labels
3. Create single partition group of P1 and P2 of GP1 and GP2 respectively
4.For i=1to n
Read edge value of i^{th} column as per P1
Partitioning element from GP1is E1g
5. Calculate Euclidean value Eu of E1g
$E1u_i = Eu$
6. Sort the values of E1u array
7. For i=1to n
Read edge value of i^{th} column as per P2
partitioning element from GP2 is E2g
8. Calculate Euclidean value Eu of E2g
$E2u_i = Eu$
9. Sort the values of E2u array
10. Check E1u = E2u
11. Detect graph are isomorphic
end

Algorithm2: Modified Algorithm for graph isomorphism using Euclidean Vector

V. EXPERIMENTAL RESULTS

We performed number of experiments on randomly generated graphs. Experiments were carried out for both directed and undirected graph .The experiment environment is: Intel(R) Core(TM) i3 CPU 540 @ 3.07GHz, Speed: 1,995.00 MHz, Cores: 4 1.8 GB RAM, Free memory: 426.4 MB (+ 759.7 MB Caches) Free swap: 1.8 GB with Linux (open suse 11.3) operating system.

The algorithm is implemented in c++ language. For each experiment we generated one or more model graph. We use these model graphs to create isomorphic input graphs .All the graphs generated for the experiments were directed labeled and unlabelled and undirected labeled and unlabelled.

We examine the time complexity experimentally; the time required for graph isomorphism detection using vertex invariant in the existing algorithm is more as compared to our algorithm which is implemented using vertex invariant and Euclidean vector.

For experimental reference, we named the existing algorithm as Algorithm1 and the modified algorithm as as Algorithm2. For the directed labeled graph, when the number of vertices more than 500 the existing algorithm fails to perform the number of permutations and unable to detect graph isomorphism. In our proposed Algorithm2, the Euclidean vector is computed based on permutation counts and able to detect graph isomorphism for the graph having vertices more than 1000.

Experimental result for the undirected labeled and unlabelled graph also gives better result than the existing algorithm. The existing algorithm fails to compute permutations and unable to detect graph isomorphism for the graph having vertices more than 30 in undirected labeled graph and more than 20 in unlabelled graph.

Experimental result for directed and undirected graph for existing algorithm (Algorithm 1) and our algorithm (Algorithm 2) are shown in figures. Our algorithm fails to detect subgraph isomorphism due to vertex invariant constraint

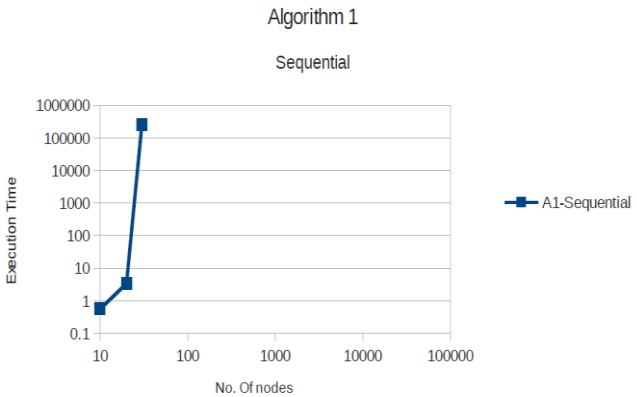


Figure 2: Time required for graph isomorphism detection for undirected labeled graph

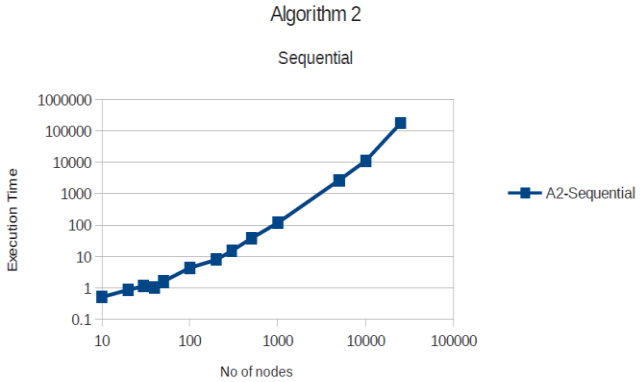


Figure 3: Time required for graph isomorphism detection for undirected labeled graph

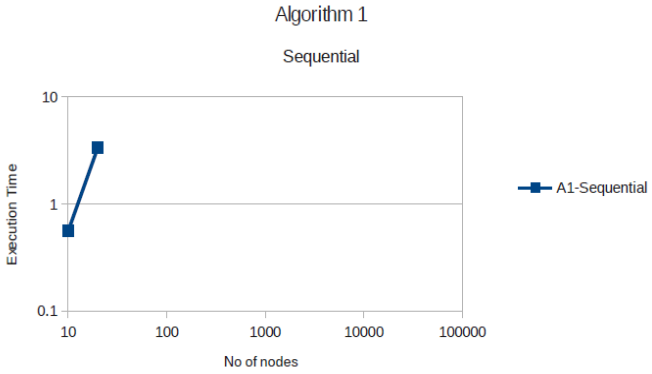


Figure 4: Time required for graph isomorphism detection for undirected unlabeled graph using algorithm A1

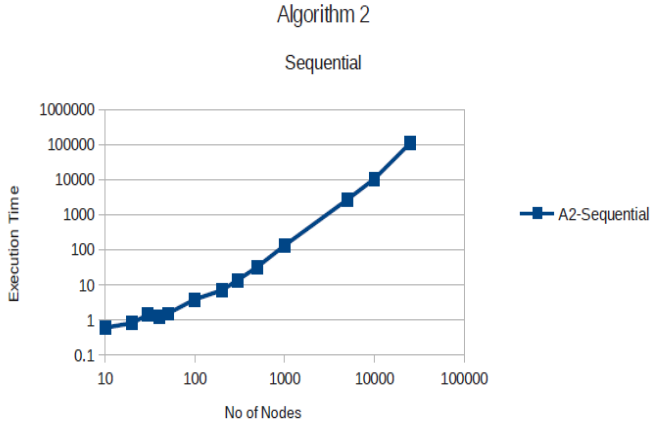


Figure 5: Time required for graph isomorphism detection for undirected unlabeled graph using algorithm A2

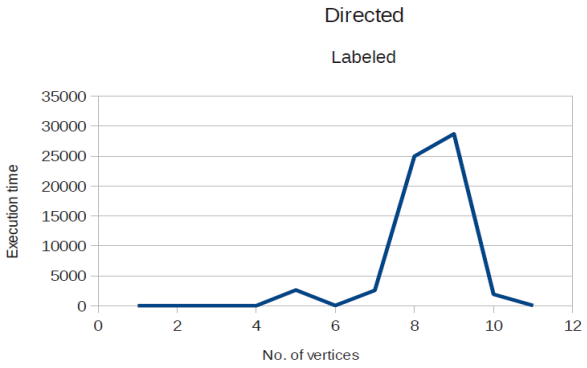


Figure 6: Time required for graph isomorphism detection for directed labeled graph using algorithm A1.

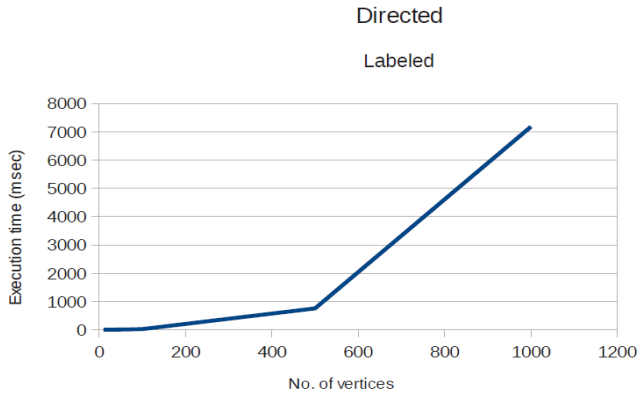


Figure 7: Time required for graph isomorphism detection for directed labeled graph using algorithm A2.

VI. CONCLUSION

We conclude that we have successfully implemented the sequential algorithm for graph isomorphism detection by using the vertex invariant approach and studied the performance with respect to time for large size randomly generated graphs. Our algorithm A2 is no longer suitable for subgraph isomorphism detection. Experimental results shows that the time complexity reduces remarkably by Algorithm2 as compared to Algorithm1for directed, undirected graph, labeled as well as unlabelled graphs.

In our next experiment we are parallelizing the algorithm to compare the performance of both algorithms.

REFERENCES

[1] T. Washio and H. Motoda. State of the art of graph-based data mining. ACM SIGKDD Explorations Newsletter, 5(1):59–68,2003

[2] D. Conte and et al. Graph matching applications in patternrecognition and image processing. In International Conferenceon Image Processing, volume 2, pages 21–24. IEEE,2003.

[3] A. T. Berztiss. A backtrack procedure for isomorphism of directed graphs. Journal of the ACM, 20(3):365–377, 1973.

[4] J. Braun and et al. Molgen-cid, a canonizer for molecules andgraphs accessible through the internet. Journal of ChemicalInformation and Computer Sciences, 44(2):542–548, 2004.

[5] Ming Qiu , Haibin Hu, Qingshan Jiang and Hailong Hu : A New Approach of Graph Isomorphism Detection based on Decision Tree IEEE, Second International workshop on Education Technology and Computer Science,2010

[6] B.T.Messmer and H.Bunke: A decision tree approach to graph and subgraph isomorphism detection Pattern Recognition 32 (1999) 1979-1998

[7] B.T.Messmer and H.Bunke: Subgraph Isomorphism in Polynomial Time. University of Bern, Institute of Computer Science and Applied Mathematics, Bern, Switzerland Technical Report IAM 1995-003, 1995

[8] Narsingh Deo: Graph Theory with Applications to Engineering and Computer Science ,Prentice Hall,Inc, 1995